An Application of Principal Component Analysis to the Detection and Visualization of Computer Network Attacks[1]

Khaled Labib and V. Rao Vemuri

Department of Applied Science
University of California, Davis
U.S.A.
kmlabib@ucdavis.edu and rvemuri@ucdavis.edu

Abstract

Network traffic data collected for intrusion analysis is typically high-dimensional making it difficult to both analyze and visualize. Principal Component Analysis is used to reduce the dimensionality of the feature vectors extracted from the data to enable simpler analysis and visualization of the traffic. Principal Component Analysis is applied to selected network attacks from the DARPA 1998 intrusion detection data sets namely: Denial-of-Service and Network Probe attacks. A method for identifying an attack based on the generated statistics is proposed. Visualization of network activity and possible intrusions is achieved using Bi-plots, which provides a summary of the statistics.

Keywords: Intrusion Detection, Principal Component Analysis, Network Traffic Visualization, Bi-plots.

I. Introduction

With the widespread use of computer networks and telecommunication devices, network security has become a primary concern for the developers and users of these networks. As a result, the problem of intrusion detection has grasped the attention of both research and corporate institutions with the aim of developing and deploying effective Intrusion Detection Systems (IDS) that are capable of protecting critical system components against intruders.

Early in the research into IDS, two major approaches known as anomaly detection and signature detection were arrived at. The former relies on flagging behaviors that are abnormal and the later flagging behaviors that are close to some previously defined pattern signature of a known intrusion [1]. This paper describes a network-based anomaly detection method for detecting Denial of Service and Network Probe attacks.

The detection of intrusions or system abuses presupposes the existence of a model [2]. In signature detection, also referred to as misuse detection, the known attack patterns are modeled through the construction of a library of attack signatures. Incoming patterns that match an element of the library are labeled as attacks. If only exact matching is allowed, misuse detectors operate with no false alarms. By allowing some tolerance in attack matching, there is a risk of false alarms, but the detector is expected to be able to detect certain classes of unknown attacks that do not deviate much from the attacks listed in the library. Such attacks are called neighboring attacks.

In anomaly detection, the normal behavior of the system is modeled. Incoming patterns that deviate substantially from normal behavior are labeled as attacks. The premise that malicious activity is a subset of anomalous activity implies that the abnormal patterns can be utilized to indicate attacks. The presence of false alarms is expected in this case in exchange for the hope of detecting *unknown* attacks, which may be substantially different from *neighboring* attacks. These are called novel attacks.

There are few other proposed models for intrusion detection that are viewed as derivatives or variants from the above two models. Specification-based intrusion detection relies on manually specifying program behavioral specification that is used as a basis to detect attacks [3]. It has been proposed as a promising alternative that combines the strengths of signature-based and anomaly-based detection. Performance

[1] A version of this paper appeared in the proceedings of SAR 2004 (http://www.hds.utc.fr/sar04)

Signature is another model proposed as a mechanism to detect anomalous program behavior indicative of a hostile attack. It is also used as a metric to guide the automated development of a patch to correct the program flaws exploited in the attack [4].

Detecting novel attacks, while keeping acceptably low rates of false alarm, is possibly the most challenging and important problem in intrusion detection.

IDSs may also be characterized by scope, as either network-based or host-based. The key difference between network-based and host-based IDSs is that a network-based IDS, although run on a single host, is responsible for an entire network, or some network segment, while a host-based IDS is only responsible for the host on which it resides [5].

In this study, a method for detecting selected types of network intrusions is presented. The selected intrusions represent two classes of attacks; namely Denial of Service (DoS) attacks and Network Probe (NP) attacks. Both attacks have a common characteristic of utilizing many packets as seen by the network interface, which are different from other attacks which may use as low as one packet to complete an attack. The method uses Principal Component Analysis (PCA) to reduce the dimensionality of the feature vectors to enable better visualization and analysis of the data. The data for both normal and attack types are extracted from the 1998 DARPA Intrusion Detection Evaluation data sets [6]. Portions of the data sets are processed to create a new database of feature vectors. These feature vectors represent the Internet Protocol (IP) header of the packets. The feature vectors are analyzed using PCA and various statistics are generated during this process, including the principal components, their standard deviations and the loading of each feature on the principal components. Bi-plots are used to represent a summary of these statistics in a graphical form. Based on the generated statistics, a method is proposed to detect intrusions with relatively low false alarm rates. The ability to visualize network activity and possible intrusions using Bi-plots are of primary importance, since they can be used by system administrators and security personnel to monitor the network traffic.

The rest of the paper is organized as follows: Section II discusses related work in intrusion detection using multivariate statistical approaches with emphasis on those using PCA. Section III provides an introduction to PCA and its applicability to the field of intrusion detection. Section IV describes Denial of Service and Network Probe attacks with emphasis on the attacks selected for this study. Section V details the process of data collection and preprocessing and the creation of feature vectors. It also describes how the various statistics are generated using PCA results. Section VI discusses the results obtained and suggests a method of detecting intrusions using these results. False alarm rates are also discussed here. Finally, Section VII provides a conclusion of the work and recommendations for future work.

II. Related Work

IDS research has been ongoing for the past 15 years producing a number of viable systems, some of which have become profitable commercial ventures [7].

There are a number of research projects that focus on using statistical approaches for anomaly detection.

Ye et al [8], [9] discuss probabilistic techniques of intrusion detection, including decision tree, Hotelling's T^2 test, chi-square multivariate test and Markov Chains. These tests are applied to audit data to investigate the frequency property and the ordering property of the data.

Taylor et al [10], [11] present a method for detecting network intrusions that addresses the problem of monitoring high speed network traffic and the time constraints on administrators for managing network security. They use multivariate statistics techniques, namely, Cluster Analysis and PCA to find groups in the observed data. They create a database of aggregated network sessions based on individual packets source and destination Internet Protocol port numbers. Then, they use PCA to reduce the dimensionality of the created vectors and overlay the cluster analysis results onto the PCA plots to see how well the cluster solution fits the natural distribution of the data points. The current study extends this work by addressing the need for better visualization of network activities and possible threats, through the use of Bi-plots. It

uses the loading values of feature vectors on the first and second principal components as measures of possible attacks. It also uses raw individual packet data to eliminate the need for extra preprocessing of the input data.

DuMouchel et al [12] discuss a method for detecting unauthorized users masquerading as registered users by comparing in real time the sequence of commands given by each user to a profile of the user's past behavior. They use a Principal Component Regression model to reduce the dimensionality of the test statistics.

Staniford-Chen et al [13] address the problem of tracing intruders who obscure their identity by logging through a chain of multiple machines. They use PCA to infer the best choice of thumbprinting parameters from data. They introduce *thumbprints*, which are short summaries of the content of a connection.

Shah et al [5] study how fuzzy data mining concepts can cooperate in synergy to perform Distributed Intrusion Detection. They describe attacks using a semantically rich language, reason over them and subsequently classify them as instances of an attack of a specific type. They use PCA to reduce the dimensionality of the collected data.

This study extends the above work by emphasizing the need for better and simple visualization tools to be used to enhance the decision making process for systems administrators and security personnel, using Bi-plots. It also suggests a simple process for data preprocessing to minimize computational costs associated with its possible implementation for real-time monitoring of traffic.

III. What is Principal Component Analysis?

Principal Component Analysis [14] is a well-established technique for dimensionality reduction and multivariate analysis. Examples of its many applications include data compression, image processing, visualization, exploratory data analysis, pattern recognition, and time series prediction. A complete discussion of PCA can be found in textbooks [15], [16]. The popularity of PCA comes from three important properties. First, it is the optimal (in terms of mean squared error) linear scheme for compressing a set of high dimensional vectors into a set of lower dimensional vectors and then reconstructing the original set. Second, the model parameters can be computed directly from the data - for example by diagonalizing the sample covariance matrix. Third, compression and decompression are easy operations to perform given the model parameters - they require only matrix multiplication.

A multi-dimensional hyper-space is often difficult to visualize. The main objectives of unsupervised learning methods are to reduce dimensionality, scoring all observations based on a composite index and clustering similar observations together based on multivariate attributes. Summarizing multivariate attributes by two or three variables that can be displayed graphically with minimal loss of information is useful in knowledge discovery. Because it is hard to visualize a multi-dimensional space, PCA is mainly used to reduce the dimensionality of d multivariate attributes into two or three dimensions.

PCA summarizes the variation in correlated multivariate attributes to a set of non-correlated components, each of which is a particular linear combination of the original variables. The extracted non-correlated components are called Principal Components (PC) and are estimated from the eigenvectors of the covariance matrix of the original variables. Therefore, the objective of PCA is to achieve parsimony and reduce dimensionality by extracting the smallest number components that account for most of the variation in the original multivariate data and to summarize the data with little loss of information.

PCA is utilized in this work as an exploratory multivariate analysis technique. Multivariate analysis is concerned with data sets that have more than one response variable for each observational unit. The data sets can be summarized by data matrices X with n rows and p columns, the rows representing the observations, and the columns the variables. The main division in multivariate methods is between those that assume a given structure, for example, dividing the cases into groups, and those that seek to discover the structure from the evidence of the data matrix alone, also called data mining. In pattern recognition terminology the distinction is between supervised and unsupervised methods. The emphasis of this paper is

primarily on using PCA as an unsupervised method with the assumption of no apriori knowledge of the structure of the input data.

In PCA, the extractions of PC can be made using either original multivariate data set or using the covariance matrix if the original data set is not available. In deriving PC, the correlation matrix may be used, instead of the covariance matrix, when different variables in the data set are measured using different units or if different variables have different variances. Using the correlation matrix is equivalent to standardizing the variables to zero mean and unit standard deviation.

The PCA model can be represented by:

$$u_{mx1} = W_{mxd}\, x_{dx1}$$

where u, an m-dimensional vector, is a projection of x - the original d-dimensional data vector ($m << d$).

It can be shown [14] that the m projection vectors that maximize the variance of u, called the principal axes, are given by the eigenvectors $e_1, e_2, \ldots, e_m$ of the data set's covariance matrix S, corresponding to the m largest non-zero eigenvalues $\lambda_1, \lambda_2, \ldots \lambda_m$.

The data set's covariance matrix S can be found from:

$$S = \frac{1}{n-1}\sum_{i=1}^{n}(x-\mu)(x-\mu)^T$$

where μ is the mean vector of x. The eigenvectors e_i can be found by solving the set of equations:

$$(S - \lambda_i I)e_i = 0 \quad i = 1,2,\ldots,d$$

where λ_i are the eigenvalues of S. After calculating the eigenvectors, they are sorted by the magnitude of the corresponding eigenvalues. Then the m vectors with the largest eigenvalues are chosen. The PCA projection matrix is then calculated as:

$$W = E^T$$

where E has the m eigenvectors as its columns. Here W is a mxd matrix.

One of the motives behind the selection of PCA for the detection of network traffic anomalies is its ability to operate on the input feature vector's space directly without the need to transform the data into another output space as in the case with self-learning techniques. For example, in Self-Organizing Maps [17], the transformation of a high-dimensional input space to a low-dimensional output space takes place through the iterative process of training the map and adjusting the weight vectors. The weight vectors are typically selected randomly which makes the process of selecting the best initial weight vectors a trial-and-error process. In PCA, dimensionality reduction is achieved by calculating the first few principal components representing the highest variance in the components of the input feature vector, without the need to perform any transformations on the input space. The input data is analyzed within its own input space, and the results of the transformations are deterministic and do not rely on initial conditions.

Another motive behind the selection of a Bi-plot as a visual approach for determining anomalies in network traffic is its simplicity and effectiveness. A simple visual approach to examine multivariate data is via a *pairs plot* or *scatterplot matrix*. Pairs plots are a set of two-dimensional projections of a high dimensional point cloud. However, a pairs plot can easily miss interesting structures in the data that depend on three or more variables, and genuinely multivariate methods explore the data in a less coordinate-dependent way. On the other hand Bi-plots can be viewed as a projection method for particular definitions of

"interestingness". Feature vectors dimensions used in this study have $d = 12$ columns, therefore, the Bi-plot visualization technique is applied to visually reduce the dimensionality of these vectors.

IV. Denial of Service and Network Probe Attacks

In a DoS attack, the attacker makes some computing or memory resource too busy, or too full, to handle legitimate users' requests. But before an attacker launches an attack on a given site, the attacker typically probes the victim's network or host by searching these networks and hosts for open ports. This is done using a sweeping process across the different hosts on a network and within a single host for services that are up by probing the open ports. This process is referred to as *Probe Attacks*.

Table I : Description of DoS and Probe Attacks (Description des attaques de type DoS et Probe considérées)

Attack Name	Attack Description
Smurf (DoS)	Denial of Service ICMP echo reply flood
Neptune (DoS)	SYN flood Denial of Service on one or more ports
IPsweep (Probe)	Surveillance sweep performing either a port sweep or ping on multiple host addresses
Portsweep (Probe)	Surveillance sweep through many ports to determine which services are supported on a single host

Table I summarizes the types of attacks used in this study. The attacks are described in more details below.

Smurf attacks, also known as directed broadcast attacks, are a popular form of DoS packet floods. Smurf attacks rely on directed broadcast to create a flood of traffic for a victim. The attacker sends a ping packet to the broadcast address for some network on the Internet that will accept and respond to directed broadcast messages, known as the Smurf amplifier. These are typically mis-configured hosts that allow the translation of broadcast IP addresses to broadcast Medium Access Control (MAC) addresses. The attacker uses a spoofed source address of the victim. For Example, if there are 30 hosts connected to the Smurf amplifier, the attacker can cause 30 packets to be sent to the victim by sending a single packet to the Smurf amplifier [18].

Neptune attacks can make memory resources too full for a victim by sending a TCP packet requesting to initiate a TCP session. This packet is part of a three-way handshake that is needed to establish a TCP connection between two hosts. The SYN flag on this packet is set to indicate that a new connection is to be established. This packet includes a spoofed source address, such that the victim is not able to finish the handshake but had allocated an amount of system memory for this connection. After sending many of these packets, the victim eventually runs out of memory resources.

IPsweep and Portsweep, as their names suggest, sweep through IP addresses and port numbers for a victim network and host respectively looking for open ports that could potentially be used later in an attack.

V. Data Collection and Preprocessing

V.1. Data Collection

The 1998 DARPA Intrusion Detection data sets were used as the source of all traffic patterns in this study. The training data set includes traffic collected over a period of seven weeks and contains traces of many types of network attacks as well as normal network traffic.

This data set has been widely used in the research in Intrusion Detection, and has been used in comparative evaluation of many IDSs. McHugh [19] presents a critical review of the design and execution of this data set.

Attack traces were identified using the time stamps published on the DARPA project web site.

V.2. Data Preprocessing

Data sets were preprocessed by extracting the IP packet header information to create feature vectors. The resulting feature vectors were used to calculate the principal components and other statistics. The feature vector chosen has the following format:

SIPx	SPort	DIPx	DPort	Prot	PLen

Where

- SIP*x* = Source IP address nibble, where *x* = [1-4]. Four nibbles constitute the full source IP address
- SPort = Source Port number
- DIP*x* = Destination IP address nibble, where *x* = [1-4]. Four nibbles constitute the full destination IP address
- DPort = Destination Port number
- Prot = Protocol type: TCP, UDP or ICMP
- PLen = Packet length in bytes

This format represents the IP packet header information. Each feature vector has 12 components corresponding to dimension *d* in the PCA equation for original data vector *x*. The IP source and destination addresses are broken down to their network and host addresses to enable the analysis of all types of network addresses.

Seven data sets were created, each containing 300 feature vectors as described above. Four data sets represented the four different attack types one each of shown in Table I. The three remaining data sets represent different portions of normal network traffic across different weeks of the DARPA Data Sets. This allows for variations of normal traffic to be accounted for in the experiment.

One of the motives in creating small data sets (i.e. 300 feature vectors each) for representing the feature vectors is to study the effectiveness of this method for real-time applications. Real-time processing of network traffic mandates the creation of small sized databases that are dynamically created from real-time traffic presented at the network interface. For real-time applications, each 300 packets arriving at the network interface are collected and preprocessed to extract the IP header information, producing 300 feature vectors that are analyzed using PCA as a single small data set. Then the process is repeated dynamically as more packets arrive at the network interface, producing a dynamic method for analyzing the traffic. Since DARPA data is only available statically, seven small data sets were created to mimic the case of dynamic real-time operation.

With each packet header being represented by a 12 dimensional feature vector, it is difficult to view this high-dimensional vector graphically and be able to extract the relationships between its various features. It is equally difficult to extract the relationship between the many vectors in a set. Therefore, the goal of using PCA is to reduce the dimensionality of the feature vector by extracting the PCs and using the first and second components to represent most of the variance in the data. It is also important to be able to graphically depict the relationship between the various feature vector components and the calculated PCs, to see which of the features affect the PCs most. This graphical representation would enable better visualization of the summary of the relationships in the data set. This visualization is achieved using Bi-plots. Note that the first two principal components only provide the direction of maximum variance in the data but do not provide a clear relationship with the traffic being normal or malicious. But the loading values of the individual feature vector components on the first two principal components indicate a strong relationship between the nature of the traffic and these loading values as will be described in section VI.

V.3. Statistics Generation

PCA was performed on all data sets where each feature vector would be represented by its 12 components. An exploratory analysis and statistical modeling tool called S-Plus [20] was used to generate the required statistics for this study. The following statistics were generated for each data set:

- Standard Deviation for each component
- Proportion of variance for each component
- Cumulative proportion of variance across all components
- Loading value of each feature on all individual components
- A Bi-plot representing the loading of the different features on the first and second components

VI. Results

The *principal component loadings* are the coefficients of the principal components transformation. They provide a convenient summary of the influence of the original variables on the principal components, and thus a useful basis for interpretation of data. A large coefficient (in absolute value) corresponds to a high loading, while a coefficient near zero has a low loading [21]. A large loading value indicates that a feature has a large influence on a given PC value. In this case, the emphasis is on the first two PCs and their loading values. The emphasis is also on those features that have the highest loading values on the two components.

The variance and standard deviation of a random variable are measures of dispersion. The variance is the average value of the squared deviation from the variable's mean, and the standard deviation is the square root of the variance.

If X is a discrete random variable with density function $f_X(x)$ and mean μ_X, the variance σ^2 is given by the weighted sum:

$$\sigma_X^2 = \sum_{i=1}^{n} (x_i - \mu_X)^2 f(x_i)$$

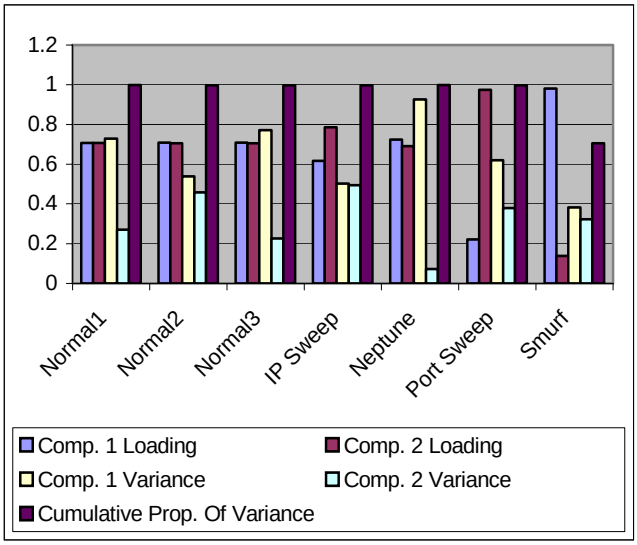


Figure 1: Component Loading and Variance (Charge des composants et variance)

Figure 1 shows the loading and variance of the first and second principal components for all data sets. Normal 1, 2 and 3 represent 3 randomly chosen data sets from normal traffic. IPsweep, Neptune, Portsweep and Smurf represent data sets for these attacks.

VI.1. Detecting Intrusions

In the results above, the first two principal components consistently had their highest absolute value loading from SPort and DPort features across all data sets with the exception of Smurf. For example, in Normal 1 data set the first two bars represent the loading values of features SPort and DPort on the first two PCs. The feature names can be seen on the Bi-plot in Figure 3 (top). The third and forth bars represent the amount of variance that each PC accounts for. The fifth bar represents the cumulative variance accounted for by both the first and second PC. It is clear that the first two PCs account for 99% of the variance in the data. And since the SPort and DPort features account for the highest loading values of these first two PCs, these features are considered to affect the variance most. This phenomena reflects the high variance in both source and destination port numbers for all data sets, except for Smurf at which the highest variance was due to source IP address components. Port numbers in TCP connections vary from 0 to 65534 and represent the different network services offered within the TCP protocol.

Note that the loading values for the first and second principal components in the three normal data sets are equal, with a value of 0.7. This represents the balance in variance in the packets flowing between a client and a server with respect to the source and destination ports. In TCP, the data and acknowledgement packets regularly flow between the client and the server, each using a designated TCP port number for the duration of the session.

For the four attack data sets, note that the loading values for the first and second principal components are not equal, possibly representing an imbalance in variance in the packets flowing between a client and a server with respect to the source and destination port numbers.

In IPsweep attacks, one or more machines (IPs) are sweeping through a list of server machines looking for open ports that can later be utilized in an attack. While in Portsweep attacks, one machine is sweeping through all ports of a single server machine looking for open ports. In both cases, there is an irregular use of port numbers that causes the variance in the PCs to vary, with an associated irregularity in the loading values.

In Neptune attacks, a flood of SYN packets is sent to one or more ports of the server machine, but from many clients with, typically, non-existing (spoofed) IP addresses. The packets seen by the server appears to be coming from many different IP addresses with different source port numbers. This is represented by the irregularity in both loading and variance of the principal components.

In Smurf attacks, attackers utilize floods of Internet Control Message Protocol (ICMP) echo reply packets to attack a server. Using amplifying stations, attackers utilize broadcast addressing to amplify the attack. The packets seen by the server appear to be coming from many different IP addresses but to one source port. Therefore, 99% of the variance for this data set is represented by the first four principal components and has their loading values associated with SIP1, SIP2, SIP3 and SIP4, instead of the source and destination ports as in previous attacks.

Figure 2 shows the standard deviation for the first and second principal components for all data sets. In the case of IPsweep and Portsweep attacks, the standard deviation of both source and destination port numbers is almost similar. This is due to the similarity in utilizing source and destination port numbers in these attacks.

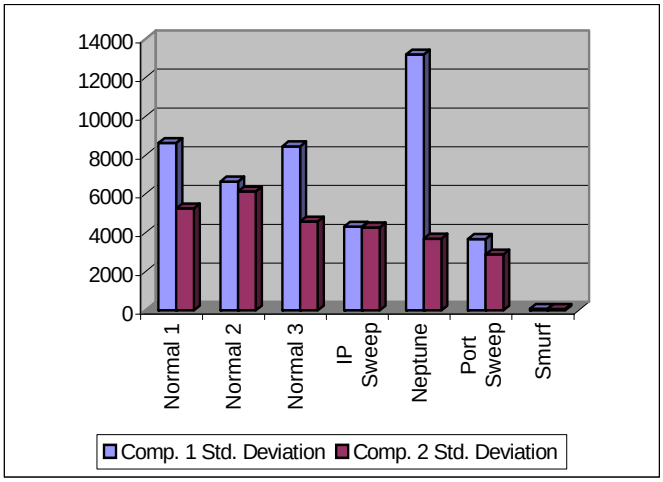


Figure 2: Standard Deviation Values for first Two Principal Components (Déviation standard estimée pour les deux premiers composants principaux)

In Neptune attacks, the source and destination ports vary differently where the source port would have the highest variance. In Smurf attacks, the first two components, namely SIP1 and SIP2, represent only a portion of the variance and have a relatively small standard deviation value.

With these results, it is possible to use the loading values of the features on the first and second principal components to identify an attack. For normal traffic, loading values appear to be similar, while during an attack the loading values differ significantly for the first two principal components. A threshold value could be used to make such a distinction. In addition, the decision could be further enhanced using the standard deviation values for first and second components. Whenever these values differ significantly, an additional data point could be obtained regarding the possibility of an attack.

Table II shows the results from a possible threshold *C* for the detection of an attack based on the loading values. This threshold is represented by the following equation:

$$C = abs\left(\left(l_1 - l_2\right)p_v * 100\right)$$

where, l_1 and l_2 are the loading values for the first and second principal components, and p_v is the cumulative proportion of variance for the first and second principal components from Figure 1.

Table II : Attack Threshold Calculation (Calcul du seuil de détection)

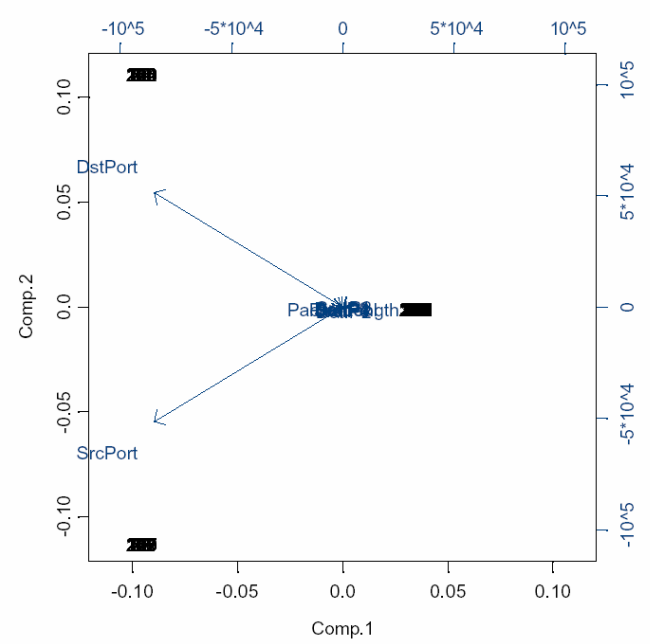
Attack Data Set	Comp. 1 Loading	Comp. 2 Loading	Cum. Prop. Of Variance	Attack Criteria
Normal 1	0.707	0.707	0.999	0.00
Normal 2	0.709	0.705	0.998	0.40
Normal 3	0.708	0.706	0.997	0.20
IP Sweep	0.617	0.787	0.998	16.97
Neptune	0.723	0.69	0.999	3.30
Port Sweep	0.221	0.974	0.998	75.15
Smurf	0.981	0.139	0.705	59.36

If a value of $C = 1$ is used given the above data sets, we could achieve a 100% detection rate. In practice, the value of C may be chosen differently to accommodate the various traffic patterns on a given network such that high detection rates and low false alarm rates are achieved. This value of $C = 1$ is an example of how the values of C should be chosen to maximize detection rates and minimize false alarms. The subset of the DARPA data sets used for this study is not by any means a representative of all network traffic that can exist in real networks.

In addition to the calculation of the attack threshold, Bi-plots could be utilized to visually interpret the loading values of the principal components and to see which features had the highest loading on a given principal component value. This could provide system administrators and security personal with a quick visual summary of the network traffic and possible intrusions.

A Bi-plot allows the representation of both the original variables and the transformed observations on the principal components axes. By showing the transformed observations, the data can be easily interpreted in terms of the principal components. By showing the variables, the relationships between those variables and the principal components can be viewed graphically.

Figure [3] shows three sample Bi-plots generated for Normal 1, Portsweep and Smurf data sets.



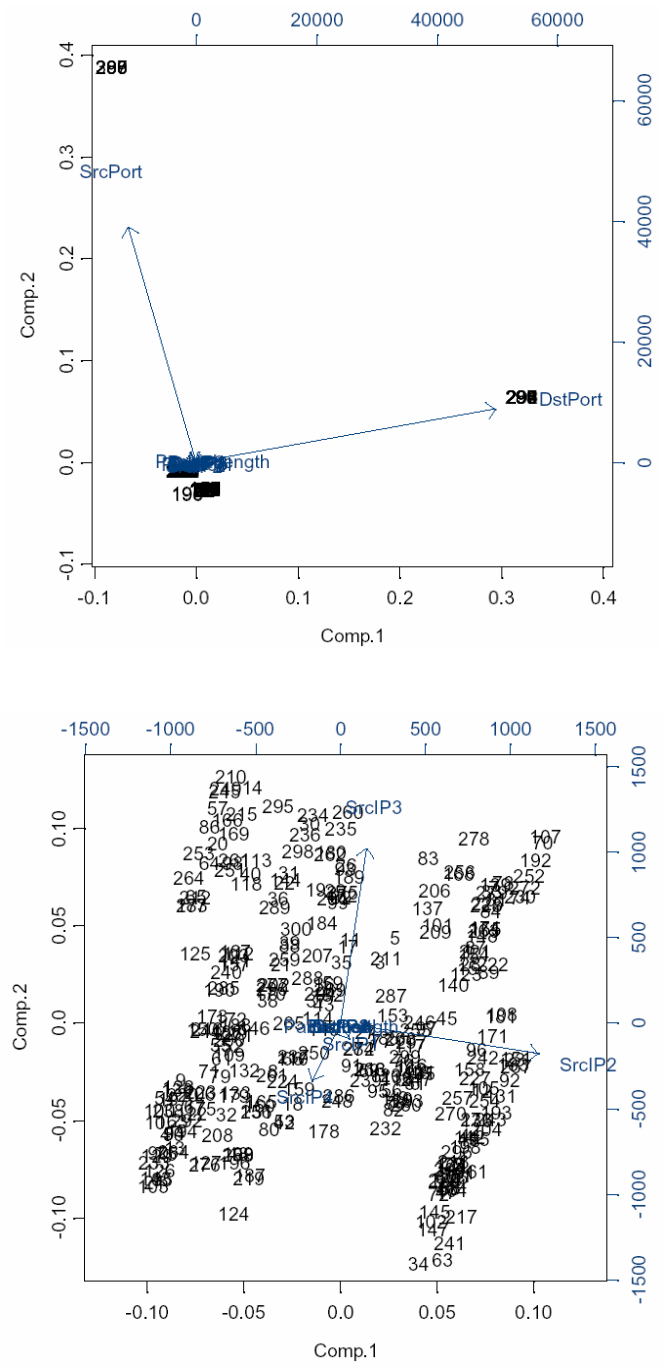


Figure 3: Bi-plots for Normal 1 (top), Portsweep (middle) and Smurf (bottom) data sets (Bi-plots pour les jeux de données normal1 (haut), portsweep (milieux) et Smurf (bas))

Interpreting the Bi-plot is straightforward: the *x*-axis represents the values for the first principal component, the *y*-axis represents the values for the second principal component. The original variables are represented by arrows, which graphically indicate the proportion of the original variance explained by the first two principal components. The direction of the arrows indicates the relative loadings on the first and second principal components.

In Figure 3, for Normal 1 data set (top), the two arrows indicate the relative loading values of features SPort (SrcPort) and DPort (DstPort) on the first two PCs. It can be seen that the *relative absolute* values are equal resulting in equal length arrows. So in normal traffic, most of the variance is due to source and destination port addresses, where the IP addresses of the source and destination remain constant within a TCP session, but the port numbers change according to the desired application or protocol each having its unique port number.

For Portsweep data set (middle), the two arrows also show that the highest loading values on the first two PCs were due to SPort and DPort, but the lengths of the vectors are not equal indicating a possible attack. Given the nature of the Portsweep attack, the attacker attempts to scan all or a subset of the ports on the destination (victim) machine looking for open ports that can be used later to compromise that machine. In doing that, the source IP and destination IP addresses remain almost constant with minimal variance, while the destination and source ports change constantly with more variance attributed to the destination address where the ports are scanned.

For Smurf data set (bottom), there are four arrows indicating that the highest loading values on the first two PCs are due to SrcIP1, SrcIP2, SrcIP3 and SrcIP4 with various arrow lengths. These four components constitute the full source IP address of a machine. A careful examination of the nature of a Smurf attack reveals that the attacker uses spoofed source IP addresses to attack a single destination (victim) machine using the Smurf amplifier as described in section IV. Therefore, most of the variance is caused by the source IP address changes. The destination IP address is constant during this attack. In addition, the variance of DPort is very minimal since this attack utilizes the ping protocol where this port is fixed. The many random numbers shown in the figure are due to the random choice of spoofed IP addresses. The Bi-plot for the Smurf data set clearly suggests the possibility of an attack given the loading values attributes and the randomness of the assignment of the original values to the plot.

With the results shown above, the Bi-plots demonstrate a simple, yet effective, graphical approach of revealing many of the PCA statistics that can be used to visualize possible anomalies in network traffic.

VI.2. Performance and Execution Time

Table III shows the execution time of the PCA algorithm including the display time for the Bi-plots graphics. Three times are shown: *User time* indicates the time (in seconds) consumed for the user process, *system time* indicates the time consumed by the operating system and the *elapsed time* indicates the total time consumed by the overall operation. The difference between user time and system time is that user time is the CPU time used while executing instructions in the user space of the calling process, while system time is the CPU time used by the system on behalf of the calling process. It should be noted that the times given include the time for applying the algorithms for all four data sets, the time to generate the Bi-plots graphics and finally the time to write this information to the disk.

Algorithm/Time	User (s)	System (s)	Elapsed (s)
PCA / Bi-plot	1.28	0.03	1.59

Table III : Algorithm Execution time (Le temps d'exécution d'algorithme)

The data in Table III is collected using a personal computer running Linux operating system with a Pentium 2 CPU running at 266Mhz and a low-end graphics controller. With a total elapsed time of 1.59 seconds used in processing 1200 packets, the method can process roughly 1000 packets per second. These performance numbers may not be adequate in handling real-time network traffic. Utilizing a more powerful computer system with advanced graphics hardware for executing the algorithm and displaying the Bi-plots can further enhance the performance of the method. In addition, the libraries used to load the packet data from the disk and the graphics routines used for display can further be optimized to achieve higher performance numbers.

VII. Conclusion and Future Work

This study presents a method for detecting Denial-of-Service attacks and Network Probe attacks using Principal Component Analysis as a multivariate statistical tool. The study described the nature of these attacks, introduced Principal Component Analysis and discussed the merits of using it for detecting intrusions. The study presented the approach used to extract the Principal Components and the related statistics. It also discussed the results obtained from using a proposed threshold value for detecting the subject intrusions. This threshold value can yield 100% detection rate using the limited sets of data used. The study presented a graphical method for interpreting the results obtained based on the Bi-plots. Future work includes testing this model to work in a real-time environment at which network traffic is collected, processed and analyzed for intrusions dynamically. This may involve using a more comprehensive criterion that accounts for other statistics including standard deviation values of the Principal Components. In addition, an enhancement may be added to utilize Bi-plots for visual interpretation of data in real-time. In this case the entire DARPA data sets can be used to qualify the results.

VIII. References

[1] Axelsson (S.), Intrusion Detection Systems: A Survey and Taxonom. *Technical report 99-15, Dept. of Computer Engineering, Chalmers University of Technology*, Goteborg, Sweden, March 2000.

[2] Cabrera (J.), Ravichandran (B.), Mehra (R.), Statistical Traffic Modeling for Network Intrusion Detection. Proceedings of the *Eighth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, Aug. 2000

[3] Uppuluri (P.), Sekar (R.), Experiences with Specification-Based Intrusion Detection. *Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection*, 2001, pp. 172 – 189

[4] Oppenheimer (D.) and Martonosi (M.), Performance Signatures: A Mechanism for Intrusion Detection. *Proceedings of the 1997 IEEE Information Survivability Workshop*, 1997.

[5] Shah (H.), Undercoffer (J.), Joshi (A.), Fuzzy Clustering for Intrusion Detection. *FUZZ-IEEE*, 2003

[6] DARPA Intrusion Detection Evaluation Project: http://www.ll.mit.edu/IST/ideval/

[7] Allen (J.) et al, State of the Practice: Intrusion Detection Technologies. *Carnegie Mellon, SEI, Tech. Report* CMU/SEI-99-TR-028, ESC-99-028, January 2000

[8] Ye (N.), Li (X.), Chen (Q.), Emran (S.), Xu (M.), Probabilistic Techniques for Intrusion Detection Based on Computer Audit Data. *IEEE Transactions on Systems, Man and Cybernetics – Part A: Systems and Humans*, Vol. 31, No. 4, July 2001

[9] Ye (N.), Emran (S.), Chen (Q.), Vilbert (S.), Multivariate Statistical Analysis of Audit Trails for Host-Based Intrusion Detection. *IEEE Transactions on Computers*, Vol. 51, No. 7, July 2002

[10] Taylor (C.), Alves-Foss (J.), NATE: Network Analysis of Anomalous Traffic Events, a low-cost approach. *NSPW'01*, September 10-13th, 2002, Cloudcroft, New Mexico, U.S.A.

[11] Taylor (C.), Alves-Foss (J.), An Empirical Analysis of NATE – Network Analysis of Anomalous Traffic Events. *New Security Paradigms Workshop'02*, September 23-26, 2002, Virginia Beach, Virginia.

[12] DuMouchel (W.), Schonlau (M.), A Comparison of Test Statistics for Computer Intrusion Detection Based on Principal Component Regression of Transition Probabilities. Proceedings of the *30th Symposium on the Interface: Computing Science and Statistics*. Fairfax Station, VA, March 6, 2000

[13] Staniford-Chen (S.), Heberlein (L.T.), Holding Intruders Accountable on the Internet. Proceedings of the *Seventh ACM Conference on Computer and Communications Security.*

[14] Hotelling (H.), Analysis of a Complex of Statistical Variables into Principal Components. *Journal of Educational Psychology*, 24:417–441, 1933.

[15] Duda (R.), Hart (P.), Stork (D.), Pattern Classification. Second Edition, *John Wiley & Sons, Inc.*, 2001

[16] Haykin (S.), Neural Networks: A Comprehensive Foundation. Second Edition. *Prentice Hall Inc.*, 1999

[17] Kohonen (T.), Self-Organizing Maps. New York, *Springer-Verlag*, 1995.

[18] Skoudis (E.), Counter Hack: A Step-by-Step Guide to Computer Attacks and Effective Defenses. *Prentice Hall Inc.*, 2002

[19] McHugh (J.), Testing Intrusion Detection Systems: Critique of the 1998 DARPA Intrusion Detection System Evaluations as Performed by Lincoln Laboratory. *ACM Transactions on Information and System Security*, Vol. 3, No. 4, November 2000, Pages 262-294

[20] http://www.insightful.com/

[21] S-Plus: Guide to Statistics, Volume 2. *Insightful Corporation*, 2001